# Stock Prediction w/ Sentiment Analysis

## Team Members:

- Kemuel Garcia Perez and Edmund Hwang

## Summary

For our project, we decided on a stock prediction system that leverages machine learning models and is enhanced with sentiment analysis of financial news headlines. The system utilizes a hybrid cloud-edge architecture to balance computational needs, with edge computing handling data analysis and prediction while cloud computing manages sentiment analysis and storage. By incorporating news sentiment as a bias in traditional technical analysis, the system aims to provide more accurate stock movement predictions.

## Problem Statement

Traditional stock prediction systems face several significant challenges:

1. **Data Overload**: Financial markets generate vast amounts of data that require real-time processing and intelligent analysis.
2. **Latency Issues**: Centralized cloud-based models introduce delays in data processing and execution, critical in high-frequency trading environments.
3. **Pattern Recognition Limitations**: Extracting insights from raw market data using traditional techniques is difficult due to market volatility, noise, and varying information sources.
4. **Sentiment Impact:** Technical indicators fail to capture market sentiment from news, social media, and other external factors influencing stock prices.
5. **Security and Privacy Concerns:** Trading data is sensitive, and using centralized cloud platforms may introduce vulnerabilities.
6. **Resource Constraints:** Advanced AI models require high computational resources, making deployment challenging in real-time trading environments.

## Solution Summary

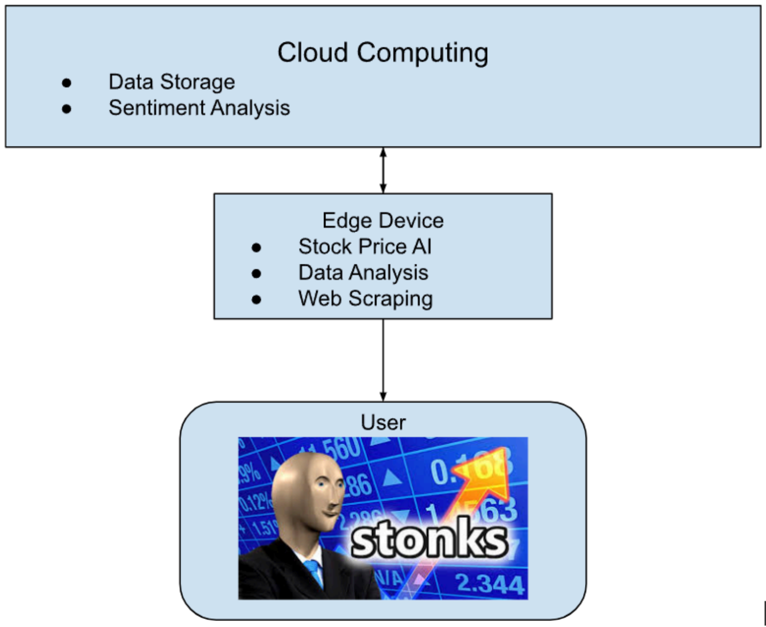
Our solution is an Edge-AI/ML Stock Prediction System that:

1. **Combines Technical and Sentiment Analysis**: Integrates traditional technical indicators with news sentiment to provide a more holistic prediction model.
2. **Implements Hybrid Architecture**: Utilizes edge computing for data collection and analysis to reduce latency while utilizing cloud computing for storage and complex sentiment analysis.
3. **Uses Automation**: From data collection to visualization, the system automates scraping news headlines, calculating sentiment scores, etc.
4. **Provides Interpretable Results**: Creates visualizations showing both the technical prediction and the impact of sentiment on these predictions.

## System Architecture

The system consists of 2 main components:

- **Google Cloud Platform** (Cloud Services)
  - Provides Natural Language Processing (NLP) for sentiment analysis
  - Offers scalable storage for sentiment data and models
  - Handles computationally intensive language processing task
- **Jeston Nano** (Edge Device)
  - Handles local computation, data preprocessing, and prediction models
  - Responsible for web scraping, technical analysis, and visualization
  - Runs the main application logic and ML prediction algorithms



## System Considerations

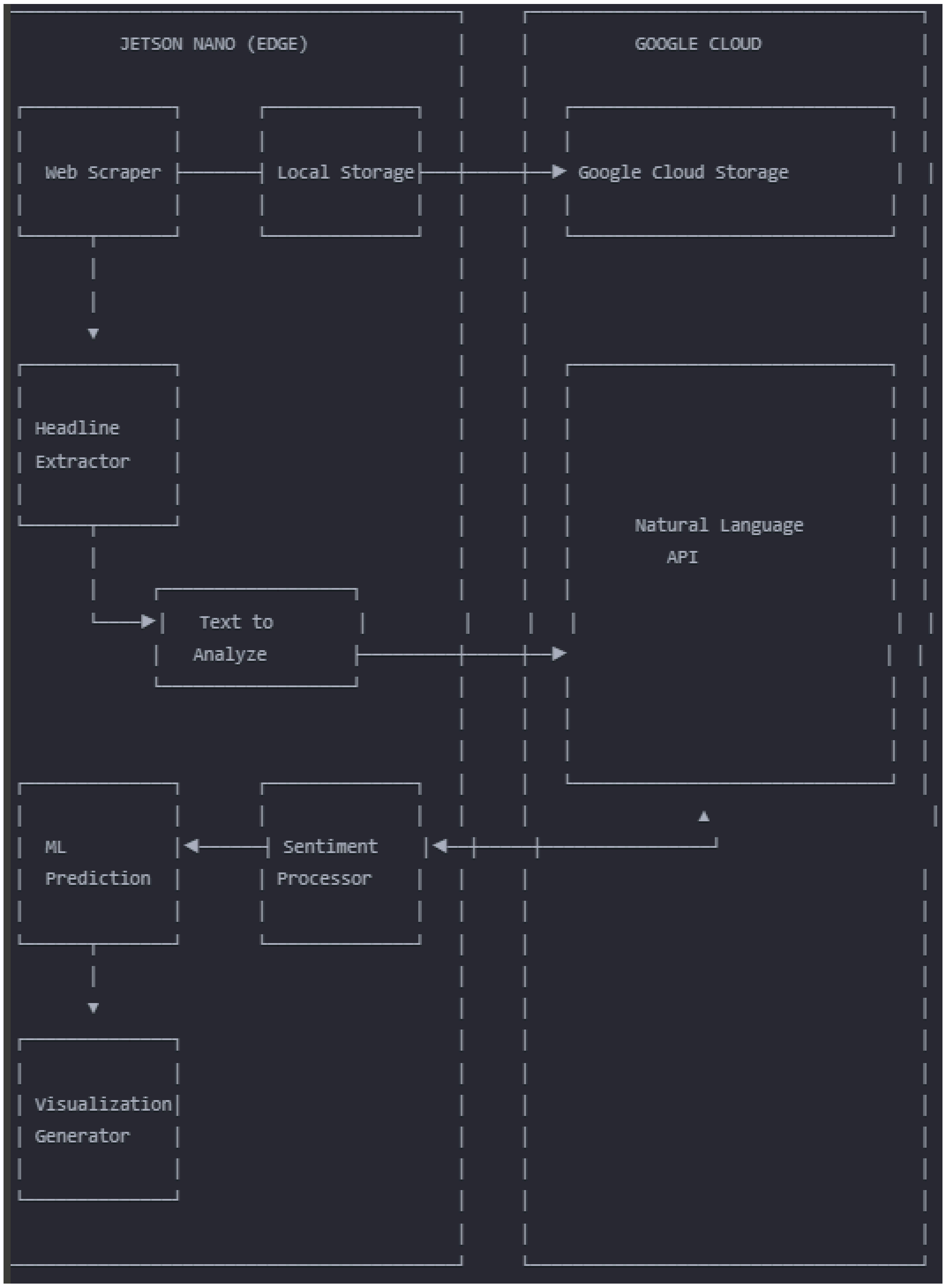
- **Minimizing API Calls**: Batching multiple headlines for sentiment analysis when possible
- **Local Caching**: Storing sentiment results locally to avoid redundant API calls
- **Selective Cloud Storage**: Only uploading essential data to Google Cloud
- **Error Handling**: Implementing robust error handling for network issues
- **Security Considerations**
    - **Service Account Authentication**: Using a dedicated service account with minimal required permissions
    - **HTTPS Encryption**: All API calls use encrypted HTTPS connections
    - **Access Control:** Cloud storage buckets configured with appropriate access controls
    - **Error Logging**: Monitoring and logging all communication errors
- **Deployment Considerations** (Things to keep in mind/needed for the system)
    - **Network Reliability**: Ensure the Jetson Nano has stable internet connectivity
    - **Quota Management**: Monitor Google Cloud API quotas to avoid being charged money
    - **Cost Optimization**: Balance cloud and edge processing to manage operating costs
    - **Service Account Security**: Protect the service account key and change it periodically
    - **Bandwidth Usage**: Monitor data transfer volumes, especially when scaling up significantly

## Communication Flow

1. Authentication and Setup
    a. The Jetson Nano authenticates with Google Cloud using a service account key file. This establishes secure communication channels for both:
        i. Google Cloud Natural Language API
        ii. Google Cloud Storage
2. Data Collection (Edge Processing)
    a. The Jetson Nano performs web scraping of financial news from Finviz.com
3. Cloud Communication for Sentiment Analysis

    a. When headlines are collected, the system sends text data to Google Cloud for sentiment analysis
    b. This is done through API calls that:
        i. Send the headline text to Google Cloud Natural Language API
        ii. Receives back sentiment scores (positive/negative) and magnitude
        iii. Handles any communication errors
4. Data Storage in Google Cloud
    a. The system uses Google Cloud Storage for persistent data storage
    b. The data is uploaded to Google Cloud Storage as such:
        i. Raw news data (JSON format)
        ii. Sentiment analysis results
        iii. Trained prediction models (pickle files)
        iv. Final prediction outputs
5. Data Retrieval from Google Cloud
    a. When needed, the system retrieves data from Google Cloud
    b. This allows the system to:
        i. Retrieve previously-stored sentiment data
        ii. Access trained models from previous runs
        iii. Enable continuity between system restarts

## Data Flow Diagram



## Technical Implementation

- **Data Collection and Sentiment Analysis**
    - The system collects data from two primary sources:
        - **Historical Stock Data**: Using the Yahoo Finance API (yfinance), the system downloads 60 days of price data for selected stocks, including open, high, low, close prices, and volume.
        - **Financial News Headlines**: The system scrapes news headlines from Finviz.com using requests and BeautifulSoup libraries, extracting both the headline text and the associated ticker symbols.
    - For sentiment analysis, the system uses Google Cloud's Natural Language API, which provides sentiment scores ranging from -1 (very negative) to +1 (very positive), along with a magnitude score indicating the strength of the sentiment.
- **Feature Engineering and Model Training**
    - For each stock, the system calculates several technical indicators:
        - Moving averages (5-day, 10-day, 20-day)
        - Daily and 5-day returns
        - Volatility (standard deviation of returns)
        - Volume changes
        - Relative Strength Index (RSI)
    - These indicators serve as features for a Random Forest classifier that predicts whether the stock price will move up or down in the next trading session. The model is trained on historical data with a target variable indicating whether the closing price increased the following day

## Technical Components and Libraries

- **test.py - Main System Implementation**
    - This script contains the core functionality of the system, including:
    - **GCSHandler class**: Manages Google Cloud Storage operations
        - Libraries: google.cloud.storage
        - Purpose: Uploads and downloads data to/from Google Cloud Storage
    - **SentimentAnalyzer class**: Handles web scraping and sentiment analysis
        - Libraries: requests, BeautifulSoup, google.cloud.language_v1

        - Purpose: Scrapes news headlines, performs sentiment analysis, and stores results
    - **StockPredictor class:** Manages stock data and prediction models
        - Purpose: Downloads stock data, calculates features, trains models, and makes predictions
        - **Libraries**:
            - Scikit-learn (**sklearn**)
                - **sklearn.ensemble.RandomForestClassifier**: Creates the prediction model for stock movements
                - **sklearn.preprocessing.StandardScaler**: Normalizes feature data for better model performance
            - **Pickle**
                - Saves trained machine learning models to files
                - Saves data preprocessing objects (like scalers)
                - Enables model persistence between program run
            - **Requests**
                - Fetches web pages from financial news sites (Finviz.com)
                - Handles HTTP sessions with appropriate headers
            - **BeautifulSoup**
                - Parses HTML from financial news websites
                - Extracts specific elements like headlines and ticker symbols
                - Navigate the DOM structure to find relevant data
            - **Google.cloud.storage**
                - Uploads sentiment data and predictions to cloud storage
                - Downloads previously stored data when needed
                - Manages cloud storage buckets and blobs
            - **Google.cloud.language.v1**
                - Analyzes sentiment of news headlines
                - Extracts sentiment scores (positive/negative) and magnitude
                - Processes natural language text
            - **OS**

            - Creates directories for data storage
            - Handles file paths across different operating systems
            - Sets environment variables for cloud authentication
            - Checks if files exist before operations
        - **Logging**
            - Creates directories for data storage
            - Handles file paths across different operating systems
            - Sets environment variables for cloud authentication
            - Checks if files exist before operations
        - **Datetime**
            - Calculates date ranges for historical data
            - Handles time-based operations
            - Timestamps predictions and analysis results
    - **Outputs:**
        - **sentiment_scores.json**: Contains sentiment analysis results for each stock
        - **stock_predictions.json:** Contains prediction results with sentiment influence
        - Trained **model files (.pkl)** for each stock
- simplified-viz.py - Visualization Generator
    - This script handles all visualization aspects of the system:
    - **Libraries**:
        - Pandas (pandas) for data manipulation and analysis library
            - Creates and manipulates DataFrames to store stock price data
            - Calculates financial indicators like moving averages and returns
            - Handles time series data with datetime indexing
            - Provides data structures for ML model training
        - **Numpy**
            - Performs mathematical operations on stock data arrays
            - Creates target variables for model training (up/down classification)
            - Sorts and manipulates array data for visualizations
            - Provides efficient numerical operations
        - **Matplotlib**

            - Creates stock price charts with technical indicators
            - Generates bar charts for sentiment comparison
            - Produces visual summaries of prediction results
            - Handles date formatting on charts
        - **yfinance**
            - Downloads historical stock price data
            - Retrieves OHLC (Open, High, Low, Close) and volume data
            - Gets corporate actions data (dividends, splits)
    - **Purpose**: Creates various charts and visualizations based on prediction and sentiment data
    - **Key outputs:**
        - Individual stock charts (e.g., AAPL_prediction_simple.png)
        - Comparison visualizations (model_vs_sentiment.png, sentiment_impact.png)
        - Summary visualizations (prediction_summary.png, headline_sentiment.png)

## Visualization Generation

The system generates several types of visualizations:

- **Individual Stock Charts**: Shows historical prices, moving averages, and prediction details.
- **Model vs. Sentiment Comparison**: Compares prediction probabilities before and after sentiment adjustment.
- **Sentiment Impact Chart:** Visualizes how much sentiment influenced each stock's prediction.
- **Prediction Summary:** Provides an overview of all predictions with sentiment indicators.
- **Headline Sentiment Summary**: Shows average sentiment scores for each stock.

## Results and Analysis

The system was tested on major tech stocks including AAPL, MSFT, GOOGL, AMZN, META, NVDA, and TSLA. The results demonstrate several key insights:

1. **Sentiment Impact Varies**: Different stocks show varying levels of sentiment impact, with some heavily influenced by news sentiment while others remain largely driven by technical factors.
2. **Headline Volume Matters**: Stocks with more headlines tend to have stronger sentiment biases, highlighting the importance of data volume in sentiment analysis.
3. **Technical-Sentiment Balance:** The system effectively balances technical indicators with sentiment analysis, providing a more nuanced prediction than either approach alone would offer.
4. **Visualization Enhances Interpretation:** The visualization tools greatly improve the interpretability of predictions, making it easier to understand why certain predictions were made and how sentiment influenced them.

## Challenges Encountered

1. **Web Scraping Complexity**: Finviz.com's structure required careful parsing to extract headlines and associated tickers accurately.
    a. NOTE: Another issue that arose was that when the site's HTML structure is updated, the scraper needs to be updated as well
2. **API Limitations**: Google Cloud API rate limits and costs need to be managed efficiently.
    a. NOTE: We also struggled with setting up communications with the API's and our local code/files
3. **Data Quality Issues**: Some stocks had limited news coverage, resulting in insufficient sentiment data.
4. **Model Tuning**: Balancing the influence of sentiment versus technical indicators required careful calibration.
5. **Visualization Design**: Creating informative yet concise visualizations that show both technical and sentiment factors was challenging
6. **Python:** It was our first time working with python on this scale which was difficult at first but with the help of Youtube, GitHub, Google, AI, and documentation we were able to get it done.

## Future Improvements

Several improvements could enhance the system's capabilities:

1. **Expanding Data Sources**: Incorporating social media sentiment, SEC filings, earnings reports, and other alternative data could provide more comprehensive sentiment analysis.
2. **Advanced NLP Techniques**: Using more sophisticated NLP models could better understand the context and implications of financial news.
3. **Real-Time Processing**: Introducing a way to have the system automatically scrape, analyze, and update reports/storage would make the system more responsive to breaking news.
    a. NOTE: we were able to successfully automate the scraping process but struggled to automate updating the stored files in an optimal manner
4. **Multiple Timeframe Prediction**s: Extending predictions to different time horizons (short, medium, and long-term) would provide more versatile trading signals.
5. **Model Optimization**: Implementing more advanced machine learning techniques, including deep learning models for both technical and sentiment analysis.
6. **User Interface**: Developing a web or mobile interface for easier interaction with the system.

## Conclusion

The Stock Prediction with Sentiment Analysis system demonstrates the potential of combining traditional technical analysis with modern sentiment analysis in a hybrid edge-cloud architecture. By leveraging the strengths of both approaches, the system provides more nuanced predictions than either method alone could achieve.

The modular design, comprehensive visualization tools, and integration of Google Cloud services create a flexible and powerful platform that could be further extended to incorporate additional data sources and more sophisticated models. As financial markets continue to be influenced by both technical factors and public sentiment, such hybrid approaches represent a promising direction for future trading systems.

## Live Demo

- Attached as a mp4 file

7.